THE JOURNAL OF NAVIGATION (2020), **00**, 1–16. © The Royal Institute of Navigation 2020
doi:10.1017/S0373463320000247

An A*-based Bacterial Foraging Optimisation Algorithm for Global Path Planning of Unmanned Surface Vehicles

Yang Long[1,4], Zheming Zuo[2], Yixin Su[1] , Jie Li[3] and Huajun Zhang[1]

[1](*School of Automation, Wuhan University of Technology, Wuhan, China*)
[2](*Department of Computer Science, Durham University, Durham, UK*)
[3](*School of Computing & Digital Technologies, Teesside University, Middlesbrough, UK*)
[4](*Science and Technology College of Hubei Minzu University, Enshi, China*)
(E-mail: suyixin@whut.edu.cn)

The bacterial foraging optimisation (BFO) algorithm is a commonly adopted bio-inspired optimisation algorithm. However, BFO is not a proper choice in coping with continuous global path planning in the context of unmanned surface vehicles (USVs). In this paper, a grid partition-based BFO algorithm, named AS-BFO, is proposed to address this issue in which the enhancement is contributed by the involvement of the A* algorithm. The chemotaxis operation is redesigned in AS-BFO. Through repeated simulations, the relative optimal parameter combination of the proposed algorithm is obtained and the most influential parameters are identified by sensitivity analysis. The performance of AS-BFO is evaluated via five size grid maps and the results show that AS-BFO has advantages in USV global path planning.

KEYWORDS

1. USV. 2. AS-BFO. 3. Chemotaxis Operation. 4. A* Algorithm. 5. Sensitivity Analysis.

Submitted: 15 March 2019. Accepted: 11 April 2020.

1. INTRODUCTION. The development and application of freshwater and marine areas is becoming increasingly extensive. Owing to the special working requirements of water, most water operations need to be accomplished by ships. Unmanned surface vehicles (USVs), with the advantages of flexible controllability, strong autonomy and field operation, have been widely applied in the civil and military fields, such as maritime cruise ships, emergency rescue activities, lake patrols and hydrological monitoring (Thomas et al., 2008; Liu et al., 2015; Nad et al., 2015; Nikola et al., 2015; Liu and Bucknall, 2016; Ma et al., 2016).

In order to guide a USV through a cluttered environment, planning a high-quality and collision-free path is a critical part during the USV's voyage (Perera et al., 2015). Particularly, path planning is an important technology in the application of the USV's intelligent control and an indispensable part of driverless technology, which not only determines the

level of autonomy of the vehicle but also premises the reliability of a mission and the likelihood of success. Fundamentally, USV path planning is a branch of classical robot tracing, which mainly concerns two factors: the total path distance and safety (Zheng et al., 2014). In addition, the quality of the generated trajectory, such as smoothness and continuity, also needs to be taken into account (Smierzchalski, 1999).

Path planning technologies can be generally divided into two groups: the pre-generative approach (static planning) and the reactive approach (dynamic planning) (Liu and Bucknall, 2015). Shi et al. (2019) focused on the smoothness and seaworthiness properties of the path. A hybrid A* algorithm with motion primitive constraints is proposed to generate an initial reference path. In order to optimise the path, a number of computational intelligent algorithms have been applied. A genetic algorithm (GA) is used to determine the optimised path for a USV under environmental loads in Kim et al. (2017). The optimised paths are determined by numerical simulations. An approach of fast path planning based on a Voronoi diagram and an improved GA has been proposed in Cao (2015). Furthermore, as a branch of intelligent algorithms, the swarm intelligence algorithm plays an important role in research on USV global path planning. Song et al. (2015) proposed a method for USV global path planning based on particle swarm optimisation. By using typical obstacle modelling and the ant colony algorithm (ACO) for global path planning, the USV global path was achieved in Wang and Chi (2016). Song (2014) improved the ACO-based grid environment model for USV global path planning. Meanwhile, some works have been done to combine various intelligent swarm algorithms for USV global path planning. For example, in Hu et al. (2015), both GA and ACO are used to generate an initial pheromone distribution and carry out dynamic integration of genetic operators, which not only can improve the convergence rate of the algorithm, but also can solve the problem of precocious and poor global search ability.

The bacterial foraging optimisation (BFO) algorithm is a new bionic algorithm, which has become another hotspot in the field of heuristic computing. Owing to its advantages, such as parallel searching of the swarm intelligence algorithm and ease of jumping out from local minima, it has attracted more interest. At present, BFO has been successfully applied in many fields, such as image processing (Madhubanti and Chatterjee, 2008; Nandita et al., 2011; Rajinikanth and Couceiro, 2014), shop scheduling (Wu et al., 2007; Raj and Priya, 2013; Cheng et al., 2015; Li et al., 2015; Zhao et al., 2015), robotics (Jati et al., 2012; Mickael et al., 2012; Yang et al., 2012; Liang et al., 2013; Frantisek et al., 2014), etc.

1.1. *Contributions of this paper.* This paper proposes a more efficient grid partition-based hybrid BFO path planning method, named AS-BFO, by integrating the A* algorithm to enhance the conventional BFO algorithm, thus solving the issue of the generation of a discontinuous path. The main contributions of this paper are listed below:

(1) The bacterial foraging optimisation algorithm is improved and applied in USV global path planning under the grid environment.
(2) The cost function of the A* algorithm is integrated into the tumble motion, which solves the problem of repairing a discontinuous path.
(3) The relative optimal parameter combination is obtained and it makes the AS-BFO algorithm run effectively in different working environments.

1.2. *Organisation of this paper.* The rest of the paper is organised as follows. Section 2 describes the basic steps of the BFO. Section 3 establishes an environmental

model and introduces the AS-BFO for our problem. The parameter combination simulation and the optimal parameter combination are obtained, and the sensitivity analysis of AS-BFO parameters is carried out in Section 4. GA, ACO and AS-BFO are selected for comparison in different experimental environments. Section 5 concludes this work.

2. DESCRIPTION OF BFO.

BFO is a global random searching algorithm, whose operation aims to simulate the physiological behaviour of *Escherichia coli* bacterium in the process of foraging behaviour, the modelling iteration producing the optimal solution. The BFO consists of three principal mechanisms to find the relative optimal solution: chemotaxis, reproduction and elimination–dispersal, each of which is detailed below (Passino, 2002; Kushwaha et al., 2012; Hossain and Ferdous, 2015; Zhao and Wang, 2015).

2.1. *Chemotaxis.*

In biology, the movement of bacteria is called chemotactic behaviour, and there are two types of chemotaxis of bacteria: swim and tumble/rotate. Swimming is the motion of bacteria forwards along the same direction as the last stage; rotation is the bacteria staying in the same position and rotating itself into a new direction. These two chemotactic operations guarantee that bacteria search the problem space as well as avoid obstacles in the searching process. The new position of the bacteria after chemotaxis can be obtained by

$$\theta^i(j,k,l) = \theta^i(j+1,k,l) + C(i)\frac{\Delta(i)}{\sqrt{\Delta^{\mathrm{T}}(i)\Delta(i)}}, \tag{1}$$

where $\theta^i(j,k,l)$ represents the bacterium i in the jth chemotaxis, kth reproduction and lth elimination–dispersal step, $C(i)$ denotes the size of chemotaxis during each swim or tumble, and $\Delta(i)$ is the direction vector of the jth chemotactic step. Finally, $\Delta(i)$ is a random direction vector with a range of $[-1, 1]$.

During cell-to-cell communication, when each bacterium moves, it releases attractant to signal other bacteria to swarm towards it. At the same time, a repellent signal is also released to warn other bacteria to keep a safe distance from itself. Such a communication mechanism is also simulated by representing the combined cell-to-cell attraction and repulsion, which can be expressed by

$$\begin{aligned} J_{cc}(\theta^i(j,k,l),\theta(j,k,l)) &= \sum_{i=1}^{S} J_{cc}^i(\theta^i,\theta) \\ &= \sum_{i=1}^{S}\left[-d_{\text{attract}}\exp\left(-\omega_{\text{attract}}\sum_{m=1}^{D}(\theta_m-\theta_m^i)^2\right)\right] \\ &\quad + \sum_{i=1}^{S}\left[h_{\text{repellant}}\exp\left(-\omega_{\text{repellant}}\sum_{m=1}^{D}(\theta_m-\theta_m^i)^2\right)\right], \end{aligned} \tag{2}$$

where $J_{cc}^i(\theta^i,\theta)$ denotes the object function value, which represents a time-varying objective function, S represents the total number of bacteria, D is the number of variables to be optimised, and d_{attract}, ω_{attract}, $h_{\text{repellant}}$ and $\omega_{\text{repellant}}$ are coefficients representing the attractive depth, attractive width, repellent height and repellent width, respectively.

2.2. *Reproduction.* After the chemotaxis, the health status of each bacterium is determined by the sum of the step fitness, i.e. $\sum_{j=1}^{N_c} J(i,j,k,l)$, where N_c is the maximum step in a chemotaxis process. Based on their health status (fitness values), all bacteria are sorted. In the reproduction step, the half of the bacteria with higher fitness values survive and the others are eliminated. And then each surviving bacterium splits into two identical ones. The reproduction process keeps the population of bacteria constant.

2.3. *Elimination–dispersal.* In order to avoid the bacteria becoming stuck around the initial positions or local optima, the elimination–dispersal process is introduced in the BFO. In the elimination–dispersal process, some bacteria are selected, based on a probability P_{ed}, to be moved to another position within the environment.

3. AS-BFO FOR USV GLOBAL PATH PLANNING.

The proposed AS-BFO method is detailed in this section. In particular, given a known map with a number of obstacles, the proposed algorithm first grid partitions the given map into an $n \times n$ grid environment. Then, the conventional BFO algorithm is applied to find the optimal path. During the chemotaxis operations of the BFO, each step will be monitored to check whether or not the tumbled node is continuous with the previous node as well as the following node. If any discontinuous path is identified, the A* algorithm is employed to repair the discontinuous part. This process will be applied until an optimal solution is found. The final optimal solution will be the best path for the given map. The flowchart of the proposed method is shown in Figure 1, and the key components are detailed below.

The grid method is an effective modelling method. The method is easy to construct, modify and simulate the geographical environment. The grid map not only can simulate relatively accurate environmental information for the USV, but also can provide simulation environments of different sizes and complexity for algorithm experiments. This provides a good basis for analysing the performance of the algorithm. The electronic chart is a digital chart that is a type of map model for USV global path planning. Electronic charts can provide map information for USVs, but such maps are less flexible than grid maps. A cellular grid map is also applied to the study of global path planning. However, in cellular grid maps, the minimum steering angle can only reach 60°, so the grid map has an advantage in steering angle. USVs usually work in vast stretches of water.

The work environment can be considered as a two-dimensional space with static obstacles. Therefore, the working environment of a USV can be gridded. In the general analysis, if obstacles occupy less than one grid, it will be considered as one whole grid (Yang et al., 2012). Thus, it will establish a one-to-one correspondence between the grid number and the two-dimensional coordinates. The data structure of the algorithm is a nine-square graph centred on the current node, with a total of eight adjacent nodes. The following node must be chosen from these eight neighbours. In the $n \times n$ grid environment, the correspondence between the grid number S and its coordinates (x, y) is shown by

$$\begin{cases} x = \mathrm{mod}(S-1, n) + 0{\cdot}5, \\ y = n + 0{\cdot}5 - \mathrm{ceil}(S/n), \end{cases} \tag{3}$$

where mod(·) is the remainder operation and ceil(·) rounds an element to the nearest integer towards positive infinity. As mentioned above, the goal of path planning is to find an ordered grid with the shortest distance among feasible ordered grids.

Fig. 1 - Colour online, B/W in print

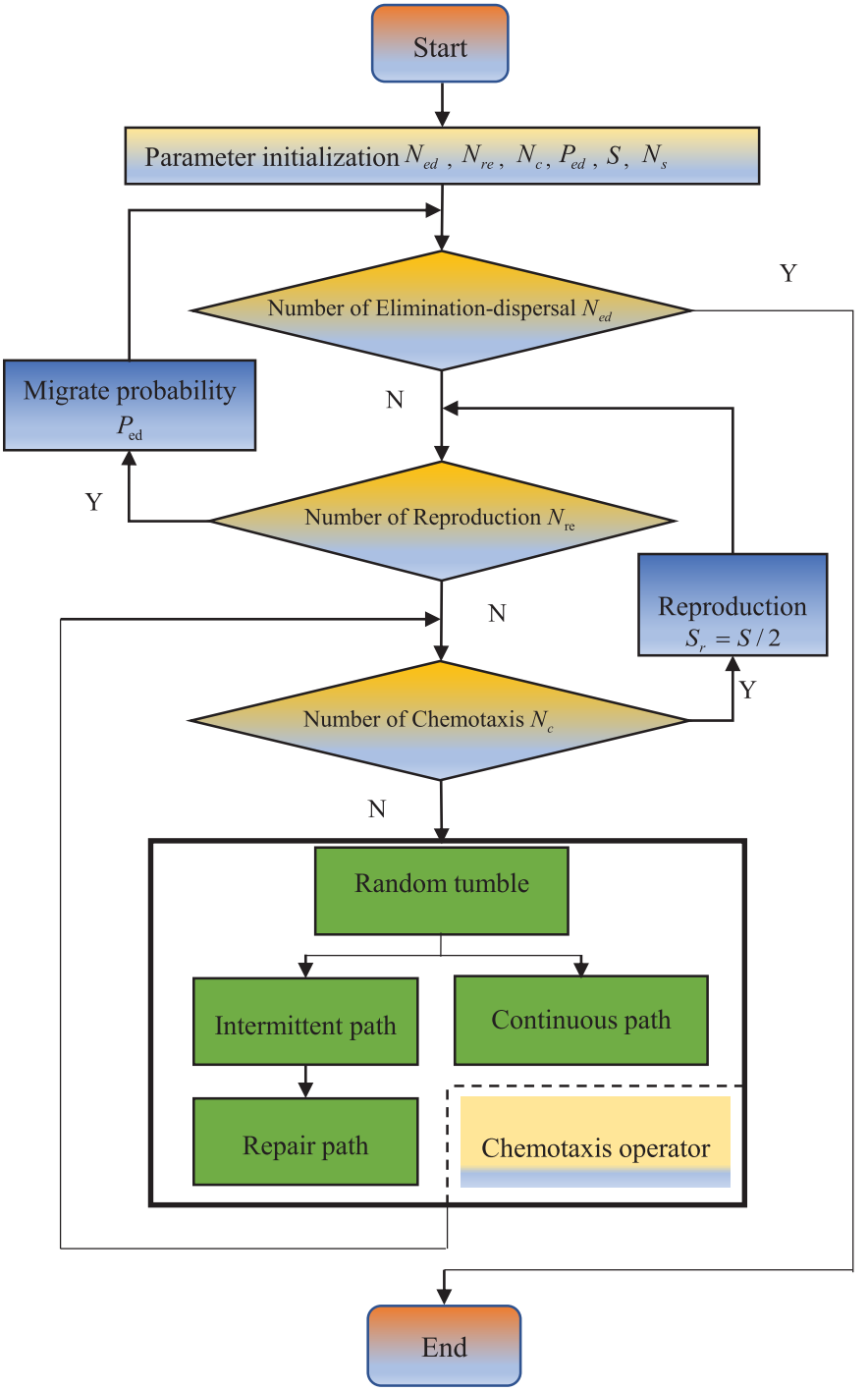

Figure 1. Flowchart of AS-BFO.

Fig. 2 - Colour online, B/W in print

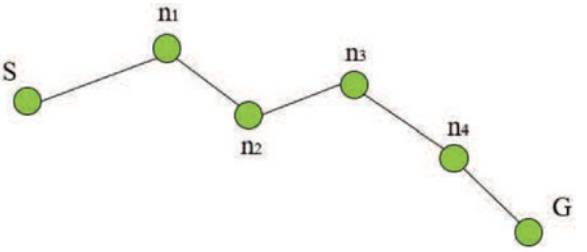

Figure 2. A feasible path.

Fig. 3 - Colour online, B/W in print

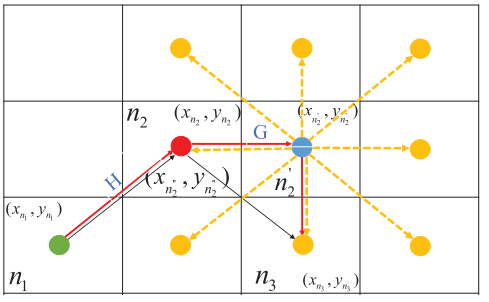

Figure 3. Schematic diagram of continuous nodes.

3.1. *Coding method.* The population described in this paper consists of a limited number of bacteria. Each bacterium is connected by a series of nodes. One bacterium represents a feasible path. It should be noted that, owing to the randomly generated paths being composed of different numbers of nodes, the lengths of the paths are not uniform. Therefore, variable length bacteria are used to represent individuals. Figure 2 shows a path and its corresponding bacterium.

3.2. *Three operators of AS-BFO.*

3.2.1. *Chemotaxis.* This paper defines the chemotaxis operation of BFO in the grid environment. Chemotaxis in traditional BFO means that any nodes will tumble and swim in any direction. However, some tumbling motions can result in discontinuous paths in the grid environment. Therefore, AS-BFO improves the chemotaxis operator. Firstly, it judges whether the tumbled nodes are continuous with the previous and the following nodes. Then, the discontinuous paths are repaired using the A* algorithm.

When the chemotaxis operator is performed, the following situations are possible. Firstly, after the node is tumbled, the tumbled node is continuous with its previous and following nodes, as shown in Figure 3. If n_2 is a tumble node, it can be randomly tumbled to the adjacent free grids. Taking Figure 3 as an example, n_2 tumbles to n_2'. Then, according to Equation (4), it is judged whether n_1 and n_2', n_2' and n_3 are, respectively, continuous. If $\Delta = 1$, then n_1 and n_2', n_2' and n_3 are continuous; otherwise, they are not:

$$\Delta = \max\{\mathrm{abs}(x_{\mathrm{tumbled}} - x)\}, \mathrm{abs}(y_{\mathrm{tumbled}} - y)\} \tag{4}$$

Here $(x_{tumbled}, y_{tumbled})$ are the horizontal and vertical coordinates of the tumbled node (n_2') and (x, y) are the horizontal and vertical coordinates of the previous or the following node (n_1, n_3). The path after the tumble motion is marked by the red line.

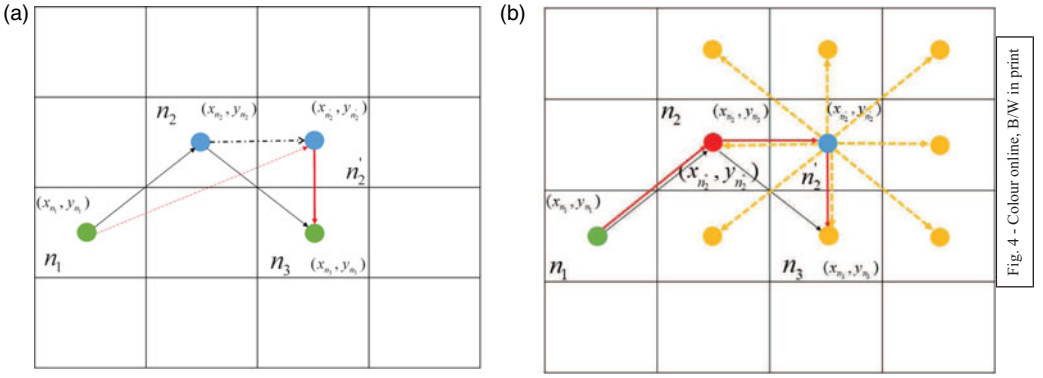

Figure 4. Schematic diagram of discontinuous nodes. (a) Intermittent path, (b) Patching path.

Secondly, it is discontinuous with the previous or the following node, when the node is tumbled. As shown in Figure 4(a), n_2 is used as the tumble node. So n_2 is tumbled to n_2', and then it is judged whether n_1 and n_2', n_2' and n_3 are, respectively, continuous. It can be observed that n_2' and n_3 are continuous; however, n_2' and n_1 are not continuous. At this time, the evaluation function of the A* algorithm is applied to the tumble motion, and the problem of repairing the discontinuous path has been solved. Figure 4(b) shows that there are many repaired nodes around $n_2'(x, y)$, and the yellow nodes are repaired nodes.

A repaired node will be selected depending on Equations (5)–(7):

$$F_{\min} = G + H, \tag{5}$$

$$G = \sqrt{(x'_{\text{tumbled}} - x''_{\text{tumbled}})^2 + (y'_{\text{tumbled}} - y''_{\text{tumbled}})^2}, \tag{6}$$

$$H = \sqrt{(x - x''_{\text{tumbled}})^2 + (y - y''_{\text{tumbled}})^2}. \tag{7}$$

Here (x, y) represents the coordinates of the previous node, $(x'_{\text{tumbled}}, y'_{\text{tumbled}})$ denotes the coordinates of a node after it has been tumbled, and $(x''_{\text{tumbled}}, y''_{\text{tumbled}})$ indicates the coordinates of the repair nodes. As shown in Figure 4(b), we use (x''_{n_2}, y''_{n_2}) as the coordinates of the repaired node. When the discontinuous path is repaired, it is represented by the red line. Although the path increases slightly, this method solves the problem that the path is discontinuous after the node is tumbled.

Finally, the tumbled node is discontinuous with the previous node and the following node after the node is tumbled. Then, the repaired point can be found by Equations (5)–(7) so that the path is continuous. Note that we need to use the evaluation function of the A* algorithm twice in this condition. The method to deal with discontinuity with the previous and the following nodes is the same as the above method. Therefore, an effective method to repair discontinuous paths is proposed, which ensures that each tumble can be performed efficiently.

3.2.2. *Reproduction.* When the chemotaxis operation is completed, the path length represented by each bacterium is sorted. The half of the bacteria with longer paths are eliminated, and the half of the bacteria with shorter paths are reproduced. Thus, bacteria number is kept unchanged.

Fig. 5 - Colour online, B/W in print

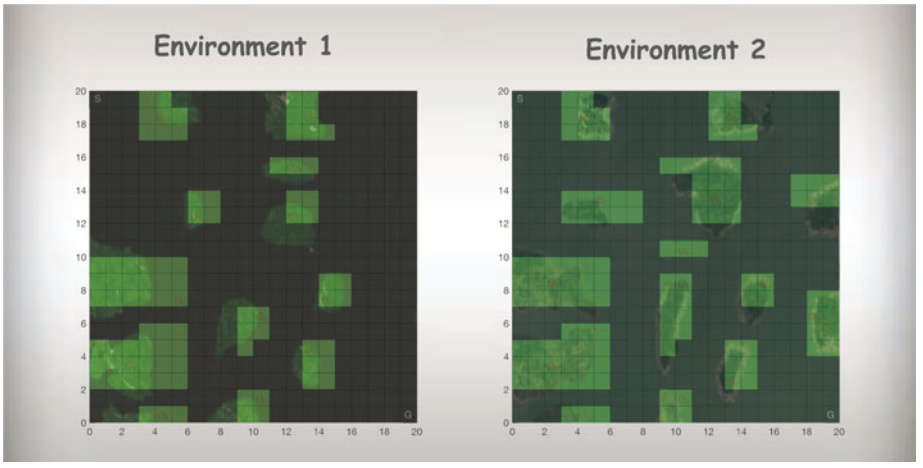

Figure 5. Generation of simulation environment in line with real ones, where S and G. The map size is 20 × 20 (i.e. S3).

3.2.3. *Elimination–dispersal.* When the reproduction operation is completed, a random probability P_r is generated and compared with the fixed migration probability P_{ed}. If $P_r < P_{ed}$, the bacterium is dispersed. This operation can reduce the possibility of bacteria falling into a local optimal solution. It can also be a good solution for maintaining diversity.

3.3. *Algorithm description.* In this algorithm, the path optimisation problem is encoded. One feasible path represents one bacterium. The initial population is usually randomly generated without infeasible paths, which can reduce the blindness of initial population generation. Among the three main operators in the algorithm, the chemotaxis operator can improve the local search accuracy of bacteria, the reproduction operator can increase the convergence performance of bacteria, and the elimination–dispersal operator can increase the diversity of solutions. The parameters of AS-BFO are tightly coupled. The selection of the parameters directly affects the performance of the algorithm. At present, there is no perfect theoretical basis to determine the optimal combination parameters. The traditional method is repeated trials to obtain the relative optimal combination of parameters.

4. ANALYSIS OF RESULTS. It is very important for the application of AS-BFO in practical problems to study the parameters of population size, chemotaxis number, replications number and elimination–dispersal number. However, the setting of the parameters mainly depends on the statistical data and simulation experiment. This paper studies the path planning in two different environments shown in Figure 5, in which the islands represent the obstacles.

4.1. *Experiment 1.* Different parameter settings may lead to different performances. In this experiment, different parameters are applied in the proposed method, in order to compare the performances generated by different parameter settings.

4.1.1. *Population number.* In this experiment, the chemotaxis number $N_c = 5$, reproduction number $N_{re} = 2$ and elimination–dispersal number $N_{ed} = 2$ are fixed, and the population number P is selected to be 10, 20, 30, 40, 50, 60, 70 and 80, respectively.

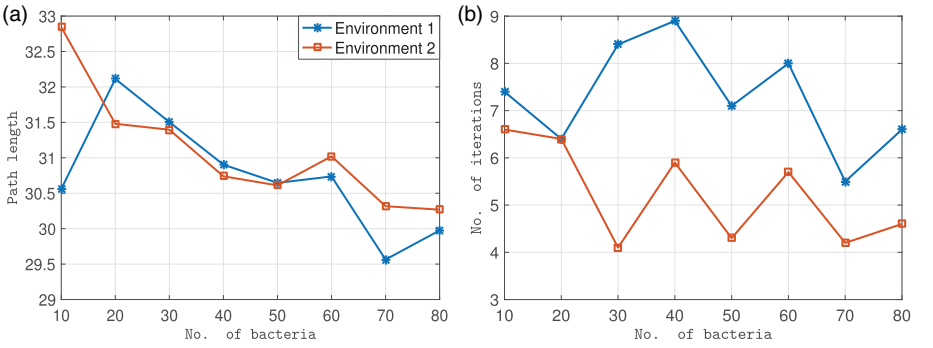

Fig. 6 - Colour online, B/W in print

Figure 6. Impact of the number of bacteria on AS-BFO using S3. (a) Correspondence between the number of bacteria and the path length. (b) Correspondence between the number of bacteria and the number of iterations.

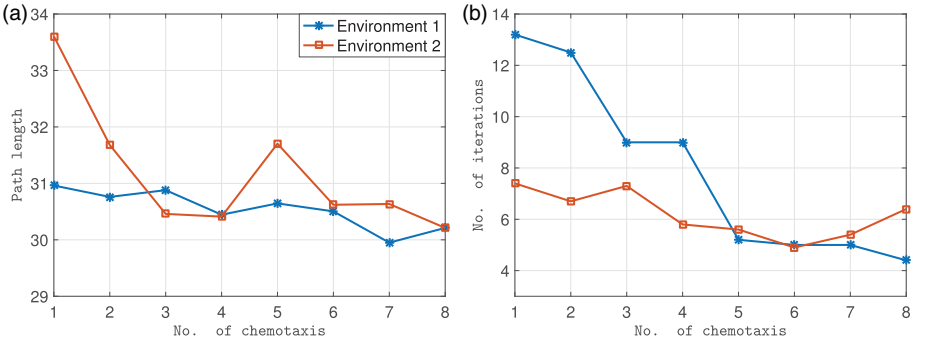

Fig. 7 - Colour online, B/W in print

Figure 7. Impact of the chemotaxis number on AS-BFO using S3. (a) Correspondence between the chemotaxis number and path length. (b) Correspondence between the chemotaxis number and number of iterations.

The simulated results are shown in Figure 6. The results show that the greater the number of bacteria, the faster the convergence rate.

4.1.2. *Chemotaxis number.* The chemotaxis operator is an important operator of AS-BFO, and the chemotaxis number directly affects the local optimisation ability of the algorithm. In this simulation experiment, $P = 50$, $N_{re} = 2$ and $N_{ed} = 2$ are fixed, and the chemotaxis number is set to $N_c = 1, 2, 3, 4, 5, 6, 7$ and 8. The results are shown in Figure 7. For environment 1, when $N_c = 7$, the path is the shortest. When $N_c = 8$, the number of iterations is the least. For environment 2, when $N_c = 8$, the path is the shortest. When $N_c = 6$, the number of iterations is the least.

4.1.3. *Reproduction number.* The reproduction operator reduces population diversity and increases the convergence rate of the algorithm. In the simulation experiment, $P = 50$, $N_c = 5$ and $N_{ed} = 2$ are fixed, and the reproduction number is set to $N_{re} = 1, 2, 3, 4$ and 5. The results are shown in Figure 8. For environment 1, when $N_{re} = 5$, the path is the shortest. When $N_{re} = 4$, the number of iterations is the least. For environment 2, when $N_{re} = 2$, the path is the shortest. When $N_{re} = 5$, the number of iterations is the least.

4.1.4. *Elimination–dispersal number.* The elimination–dispersal operator is designed to improve global optimisation and maintain the diversity of solutions. As the outermost

Fig. 8 - Colour online, B/W in print

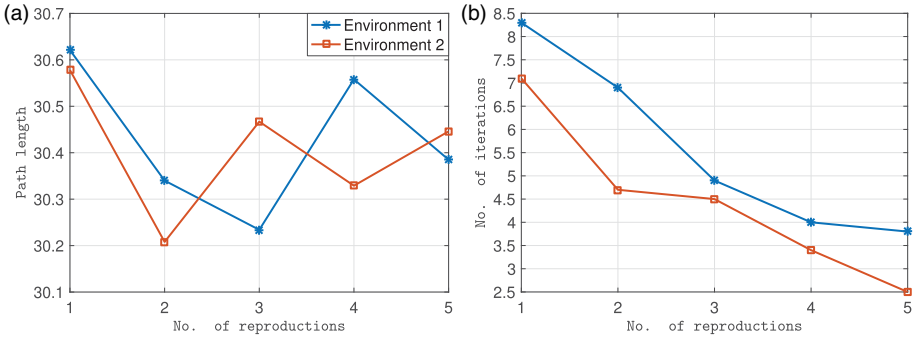

Figure 8. Impact of the number of reproductions on AS-BFO using S3. (a) Correspondence between the reproduction number and path length. (b) Correspondence between the reproduction number and number of iterations.

Fig. 9 - Colour online, B/W in print

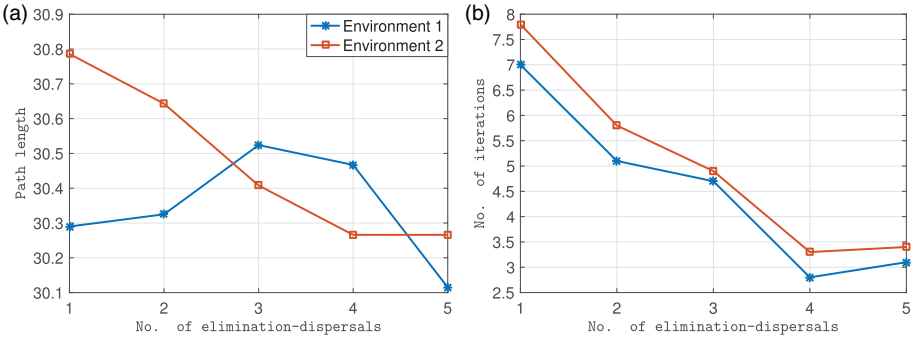

Figure 9. Impact of the number of elimination–dispersals on AS-BFO using S3. (a) Correspondence between the number of elimination–dispersals and path length. (b) Correspondence between the number of elimination–dispersals and number of iterations.

nesting of the algorithm, the elimination–dispersal number directly affects the algorithm running time. In the simulation experiment, $P = 50$, $N_c = 5$ and $N_{re} = 2$ are fixed, and reproduction number varies as $N_{ed} = 1, 2, 3, 4$ and 5. The results are shown in Figure 9. For environment 1, when $N_{ed} = 3$, the path is the shortest. When $N_{ed} = 5$, the number of iterations is the least. For environment 2, when $N_{ed} = 4$, the path is the shortest. When $N_{ed} = 4$, the number of iterations is the least.

4.2. *Parameter sensitivity analysis.* As demonstrated above, different values of parameters lead to completely different results. The most influential parameters are identified by sensitivity analysis (SA), as well as to understand their impact on the model output. For this reason, the Morris method (Cadero et al., 2018) is applied to develop the parameter sensitivity analysis. Briefly, the Morris method involves the generation of uncertain parameter samples via a trajectory-based sampling process. The method calculates and evaluates the standard deviation σ_i of the elementary effects μ_i, where i is the ith date sample, over r repetitions to assess the factors' importance. A high value of μ_i indicates a high linear effect for a given factor, while a high value of σ_i represents either nonlinear or non-additive factor behaviour. The importance of input factors of the model can often be assessed by plotting

Table 1. Sampling range of each parameter.

Parameter	Range
Population number (P)	$P \sim (10, 100)$
Chemotaxis number (N_c)	$N_c \sim (2, 12)$
Reproduction number (N_{re})	$N_{re} \sim (1, 8)$
Elimination–dispersal number (N_{ed})	$N_{ed} \sim (1, 10)$

Table 2. Extracted samples.

Factor	Trajectory 1 (P, N_c, N_{re}, N_{ed})	Trajectory 2 (P, N_c, N_{re}, N_{ed})	Trajectory 3 (P, N_c, N_{re}, N_{ed})	Trajectory 4 (P, N_c, N_{re}, N_{ed})
1	(63,2,3,7)	(100,2,8,3)	(10,5,6,10)	(30,2,1,10)
2	(63,5,3,7)	(100,9,8,3)	(10,2,6,10)	(30,5,1,10)
3	(30,5,3,7)	(100,9,8,7)	(10,2,6,1)	(63,5,1,10)
4	(30,5,3,1)	(10,9,8,7)	(30,2,6,1)	(63,5,6,10)
5	(30,5,1,1)	(10,9,6,7)	(30,2,8,1)	(63,5,6,3)

Table 3. Average optimal path of 20 data samples.

Factor	Trajectory 1	Trajectory 2	Trajectory 3	Trajectory 4
1	30·27	30·62	31·09	30·74
2	29·68	29·21	32·97	31·49
3	31·10	29·92	36·53	29·80
4	30·97	31·67	31·42	29·92
5	37·99	30·90	32·39	29·45

the factors (μ^*, σ), where μ^* is the mean of μ_i in two-dimensional space. The factors closest to the origin are less influential.

There are four uncertain parameters, population number P, chemotaxis number N_c, reproduction number N_{re} and elimination–dispersal number N_{ed}, involved in the proposed model. Table 1 shows the sampling ranges of each parameter during the sampling processes of the Morris method. In this experiment, the Morris method runs a model with five factors along four trajectories, and a total of 20 samples have been extracted, as listed in Table 2. Each generated data sample is applied as the input to the proposed method 10 times to get an average optimal path length, which is listed in Table 3. According to the Morris method, the corresponding standard deviation σ and its elementary effects μ for each trajectory can be calculated, which is shown in Figure 10. Figure 10 shows the sensitivity of each parameter in the environmental map of different sizes. In addition, the (μ^*, σ) is also plotted in a two-dimensional plot, illustrated in Figure 11, for parameter importance analysis. It is clear that parameter 3 (N_{re}) and parameter 4 (N_{ed}) are away from the origin, which indicates that those parameters are the most important.

4.3. *Algorithm comparison.* The performance of the proposed method is evaluated and validated in this section. Generally speaking, GA and ACO are adopted to against the proposed approach. Likewise, we first obtain the relative optimal parameters of ACO and GA through repeated experiments in the S3 environment, and then apply the parameters to other environments. In order to evaluate the performance of each method under different environments, the simulated environments, described in Figure 6, are grid partitioned into

Fig. 10 - Colour online, B/W in print

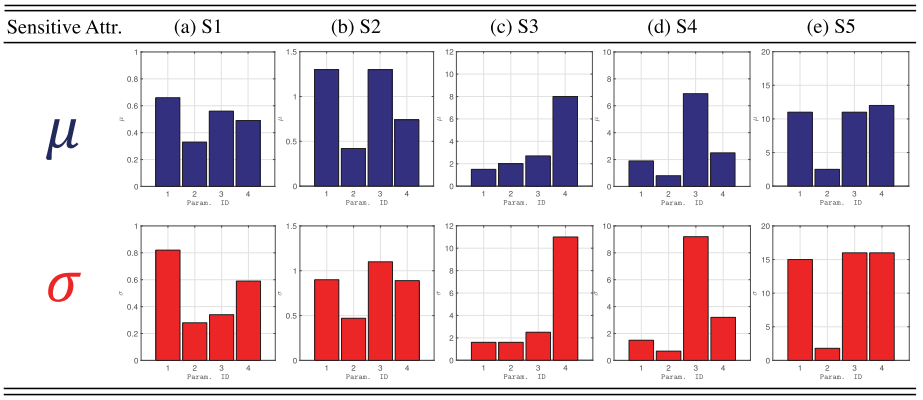

Figure 10. The results of the Morris method.

Fig. 11 - Colour online, B/W in print

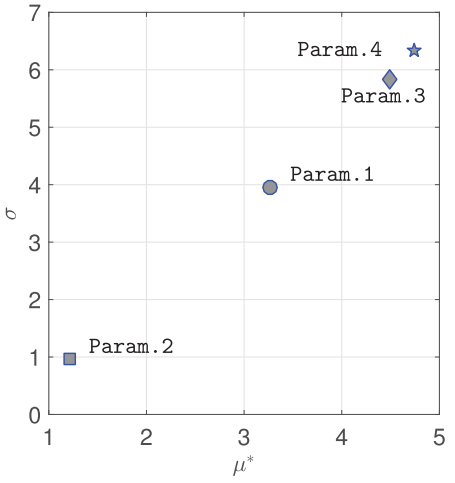

Figure 11. Screening of input factors based on the Morris method.

five different size, denoted as: S1, 10 × 10; S2, 15 × 15; S3, 20 × 20; S4, 25 × 25; and S5, 30 × 30. Each size is considered as an individual scenario. In each environment, the ratio of the obstacle area to the entire map area is approximately equal. The experimental results of three approaches under five different sizes are listed in Table 4. Figure 12 shows the relative optimal paths of the two environments in the five size maps.

From Table 4, we can see that the AS-BFO has a better average path length using a very small iteration number. Moreover, the maximum path length obtained by AS-BFO is less than ACO and GA in the 20 repeated experiments. In S1, S2 and S3, AS-BFO's AIN has a great advantage. However, the advantages in MaxPL, MinPL and APL are not obvious. In S4, the advantage of AS-BFO's AIN is still obvious. It is worth noting that AS-BFO's MaxPL is 24·39 (32·04) smaller than ACO and 6·83 (5·07) smaller than GA. AS-BFO's MinPL is 6 (14·49) smaller than ACO and 1·76 (0·59) smaller than GA. In S5, the advantages of MinPL's AIN are equally obvious. AS-BFO's MaxPL is 76·67 (53·01) smaller than ACO and 17·66 (12) smaller than GA. The MinPL of AS-BFO is 40·39 (57·93)

Figure 12. Relative optimal path of two environments in five sizes.

Table 4. Performance comparison of various optimisation methods.

		Environment 1					Environment 2				
Metric	Method	S1	S2	S3	S4	S5	S1	S2	S3	S4	S5
AIN	AS-BFO	1·2	1·3	2·3	2·8	2·8	1·0	1·8	4·6	3·4	3·6
	ACO	6·3	9·2	11·9	8·0	7·8	3·8	14·8	25·7	8·7	9·3
	GA	21·4	48·6	76·9	85·7	88·3	13·0	56·0	74·3	82·5	91·3
MaxPL	AS-BFO	13·90	21·56	30·39	40·38	50·87	14·49	22·73	31·21	41·80	48·87
	ACO	13·90	22·97	33·80	64·77	127·54	14·49	22·97	32·04	73·84	101·88
	GA	13·90	22·73	32·97	47·21	68· 53	15.07	23·56	34·39	46·87	60·87
MinPL	AS-BFO	13·31	20·97	28·62	35·11	45·11	14·49	21·56	28·62	39·21	43·95
	ACO	13·31	20·97	29·21	41·11	85·50	14·49	21·56	29·21	53·70	101·88
	GA	13·31	20·97	29·21	36·87	48·87	14·49	21·56	30·04	39·80	46·53
APL	AS-BFO	13·37	21·18	29·57	38·94	48·46	14·49	21·89	30·18	40·14	46·94
	ACO	13·40	21·46	29·95	55·13	101·65	14·49	22·14	30·89	64·30	89·43
	GA	13·61	21·73	30·46	41·54	68·53	14·51	22·72	30·95	42·66	52·93

Note: AIN, average iteration number; MaxPL, maximum path length; MinPL, minimum path length; APL, average path length. There are five map sizes used in each of the two environments: S1, 10×10; S2, 15×15; S3, 20×20; S4, 25×25; and S5, 30×30.

smaller than ACO and 3·76 (2·58) smaller than GA. Analysis shows that the larger the map size L, the greater the advantage of AS-BFO. Therefore, AS-BFO is superior to GA and ACO in searching efficiency and obtaining an optimal solution.

5. CONCLUSIONS. This paper proposes a more efficient path planning method based on the AS-BFO algorithm. The effects of bacteria number, chemotaxis number, reproduction number and elimination–dispersal number on the global path planning are analysed. Through experimental analysis, bacteria number is inversely proportional to the path length. The greater the chemotaxis number, the stronger is the local optimisation ability of the algorithm. Correspondingly, the path length and iteration number will decrease. However, the greater the reproduction number, the smaller the diversity of the population and the faster the convergence of the algorithm. The experiments indicate that the increase of elimination–dispersal number will decrease both the path length and iteration number within a certain range. It can be found from the comparison of algorithms that AS-BFO performs better than comparative algorithms in terms of average iteration number, average path, maximum path and minimum path. A parameter sensitivity analysis shows that the effects between the various parameters and the effect of each parameter on the output are different in different environmental maps.

COMPETING INTERESTS

The authors declare that there are no conflicts of interest regarding the publication of this paper.

ACKNOWLEDGEMENT

We would like to express our sincere thanks to Wuhan University of Technology (WHUT) for supporting our work.

REFERENCES

Cadéro, A., Aubry, A., Brun, F., Dourmad, J. Y., SalaĺźN, Y. and Garcia-Launay, F. (2018). Global sensitivity analysis of a pig fattening unit model simulating technico-economic performance and environmental impacts. *Agricultural Systems*, 165, 221–229.

Cao, L. (2015). Improved Genetic Algorithm for Fast Path Planning of USV. *International Symposium on Multispectral Image Processing and Pattern Recognition (MIPPR2015)*, 9815, 981529.

Cheng, Z., Tong, Y., Shen, L. and Ming, L. I. (2015). Improved bacteria foraging optimisation algorithm for solving flexible job-shop scheduling problem. *Journal of Computer Applications*, 63–67.

Frantisek, D., Babinec, A., Kajan, M., Florek, M. and Fico, T. (2014). Path planning with modified A star algorithm for a mobile robot. *Procedia Engineering*, 96, 59–69.

Hossain, M. A. and Ferdous, I. (2015). Autonomous robot path planning in dynamic environment using a new optimization technique inspired by bacterial foraging technique. *Robotics and Autonomous Systems*, 64, 137–141.

Hu, Y., Li, D. and Ding, Y. (2015). A Path Planning Algorithm Based on Genetic and Ant Colony Dynamic Integration. *IEEE Intelligent Control and Automation*, 4881–4886.

Jati, A., Singh, G. and Rakshit, P. (2012). A Hybridisation of Improved Harmony Search and Bacterial Foraging for Multi-Robot Motion Planning. *IEEE Evolutionary Computation*, 1–8.

Kim, H., Kim, S. H., Jeon, M., Kim, J. H., Song, S. and Paik, K. J. (2017). A study on path optimization method of an unmanned surface vehicle under environmental loads using genetic algorithm. *Ocean Engineering*, 142, 616–624.

Kushwaha, N., Bisht, V. S., Shah, G. and Kushwaha, N. (2012). Genetic Algorithm Based Bacterial Foraging Approach for Optimization. *IJCA Proceedings on National Conference on Future Aspects of Artificial Intelligence in Industrial Automation*, 2, 11–14.

Li, L., WU, X. and Wang, Z. (2015). Research of no-idle flow shop scheduling based on improved bacteria foraging optimization algorithm. *Computer Engineering & Applications*, 17, 048.

Liang, X., Li, L., Wu, J. and Chen, H. (2013). Mobile robot path planning based on adaptive bacterial foraging algorithm. *Journal of Central South University (English Edition)*, 20(12), 3391–3400.

Liu, Y. and Bucknall, R. (2015). Path planning algorithm for unmanned surface vehicle formations in a practical maritime environment. *Ocean Engineering*, 97, 126–144.

Liu, Y. and Bucknall, R. (2016). The angle guidance path planning algorithms for unmanned surface vehicle formations by using the fast marching method. *Applied Ocean Research*, 59, 327–344.

Liu, Y., Song, R. and Bucknall, R. (2015). A Practical Path Planning and Navigation Algorithm for an Unmanned Surface Vehicle Using the Fast Marching Algorithm. *Oceans 2015 – Genova IEEE*, 1–7.

Ma, Y., Zhao, Y., Diao, J., Gan, L., Bi, H. and Zhao, J. (2016). Design of sail-assisted unmanned surface vehicle intelligent control system. *Mathematical Problems in Engineering*, 2016, 1–13.

Madhubanti, M. and Chatterjee, A. (2008). A novel technique for multilevel optimal magnetic resonance brain image thresholding using bacterial foraging. *Measurement*, 41(10), 1124–1134.

Mickael, A., Kiani, K. and Fateh, M. M. (2012). Design of fuzzy controller for robot manipulators using bacterial foraging optimization algorithm. *Journal of Intelligent Learning Systems & Applications*, 4(1), 53–58.

Nad, D., MiSkovic, N. and Mandic, F. (2015). Navigation, guidance and control of an overactuated marine surface vehicle. *Annual Reviews in Control*, 40, 172–181.

Nandita, S., Chatterjee, A. and Munshi, S. (2011). An adaptive bacterial foraging algorithm for fuzzy entropy based image segmentation. *Expert Systems with Applications*, 38(12), 15489–15498.

Nikola, M., Nad, D. and Rendulic, I. (2015). Tracking divers: an autonomous marine surface vehicle to increase diver safety. *IEEE Robotics & Automation Magazine*, 22(3), 72–84.

Passino, K. M. (2002). Biomimicry of bacterial foraging for distributed optimization and control. *IEEE Control Systems*, 22(3), 52–67.

Perera, L. P., Ferrari, V., Santos, F. P., Hinostroza, M. A. and Guedes Soares, C. (2015). Experimental evaluations on ship autonomous navigation and collision avoidance by intelligent guidance. *IEEE Journal of Oceanic Engineering*, 40(2), 375–387.

Raj, J. S. and Priya, S. D. (2013). Contribution of BFA in Grid Scheduling. *IEEE International Conference on Computational Intelligence & Computing Research*, IEEE, 1–4.

Rajinikanth, V. and Couceiro, M. S. (2014). Multilevel Segmentation of Color Image Using Lévy Driven BFA Algorithm. *International Conference on Interdisciplinary Advances in Applied Computing*, ACM.

Shi, B., Su, Y., Wang, C., Wan, L. and Luo, Y. (2019). Study on intelligent collision avoidance and recovery path planning system for the waterjet-propelled unmanned surface vehicle. *Ocean Engineering*, 182, 489–498.

Smierzchalski, R. (1999). Evolutionary trajectory planning of ships in navigation traffic areas. *Journal of Marine Science and Technology*, 4(1), 1–6.

Song, C. H. (2014). Global path planning method for USV system based on improved ant colony algorithm. *Applied Mechanics & Materials*, 4, 568–570.

Song, L., Mao, Y., Xiang, Z., Zhou, Y. and Du, K. (2015). A study on path planning algorithms based upon particle swarm optimization. *Journal of Information & Computational Science*, 12(2), 673–680.

Thomas, S., Howells, G. and Maier, M. D. (2008). Autonomous ship collision avoidance navigation concepts, technologies and techniques. *Journal of Navigation*, 61(1), 129–142.

Wang, Y. H. and Chi, C. (2016). Research on Optimal Planning Method of USV for Complex Obstacles. *IEEE International Conference on Mechatronics and Automation*, 2507–2511.

Wang, Y., Liang, X., Li, B. and Yu, X. (2017). Research and Implementation of Global Path Planning for Unmanned Surface Vehicle Based on Electronic Chart. *International Conference on Mechatronics and Intelligent Robotics*. Springer, Cham, 534–539.

Wu, C., Zhang, N., Jiang, J. and Liang, Y. (2007). Improved Bacterial Foraging Algorithms and Their Applications to Job Shop Scheduling Problems. *International Conference on Adaptive and Natural Computing Algorithms Springer*, 562–569.

Yang, W., Wang, H. B. and Wang, J. (2012). Research on path planning for mobile robot based on grid and hybrid of GA/SA. *Advanced Materials Research*, 479-481, 1499–1503.

Zhao, F., Jiang, X., Zhang, C. and Wang, J. (2015). A chemotaxis-enhanced bacterial foraging algorithm and its application in job shop scheduling problem. *International Journal of Computer Integrated Manufacturing*, 28(10), 1106–1121.

Zhao, W. and Wang, L. (2015). An effective bacterial foraging optimizer for global optimization. *Information Sciences*, 329, 719–735.

Zheng, E. H., Xiong, J. J. and Luo, J. L. (2014). Second order sliding mode control for a quadrotor UAV. *ISA Transactions*, 53(4), 1350–1356.